Towards Practical Homomorphic Encryption with Efficient Public key Generation

Y Govinda Ramaiah, G Vijaya Kumari
Department of Computer Science and Engineering,
JNTUH College of Engineering, Hyderabad, India
Email: {ygovinda, dr.gvk}@ieee.org

***Abstract*—With the advent of cloud computing several security and privacy challenges are put forth. To deal with many of these privacy issues, 'processing the encrypted data' has been identified as a potential solution, which requires a Fully Homomorphic Encryption (FHE) scheme. After the breakthrough work of Craig Gentry in devising an FHE, several new homomorphic encryption schemes and variants have been proposed. However, all those theoretically feasible schemes are not viable for practical deployment due to their high computational complexities. In this work, a variant of the DGHV's integer based Somewhat Homomorphic Encryption (SHE) scheme with an efficient public key generation method is presented. The complexities of various algorithms involved in the scheme are significantly low. The semantic security of the variant is based on the two-element Partial Approximate Greatest Common Divisors (PAGCD) problem. Experimental results prove that the proposed scheme is very much efficient than any other integer based SHE scheme existing today and hence practical.**

***Index Terms*—Homomorphic Encryption, Implementation, Practicality, Efficient public key, Cloud security, Privacy.**

I. Introduction

A Fully Homomorphic Encryption (FHE) scheme or a privacy homomorphism [1] supports "processing the data while it is encrypted"[2][3]. The research on the topic has gained momentum after Craig Gentry's first construction of such a scheme [2][3] based on the algebraic lattice theory in the year 2009. Since an FHE scheme allows delegation of computational tasks to the remote untrustworthy server, Gentry's breakthrough work has become an attractive solution, particularly for the security and privacy problems of cloud computing and the related applications. However, existing solutions are theoretically promising, but, far away from practical implementation due to high computational complexities involved.

In simple words, an encryption scheme is said to be fully homomorphic when unlimited addition and multiplication operations are supported on the ciphertexts generated by it [2][3][6]. Such a scheme will have the capability to compute arbitrarily any function on the encrypted data. Data is encrypted bitwise, and the *circuit representation of functions* is exploited in order to evaluate a function on the ciphertexts. Apart from the two general requirements *semantic security* and *correctness* of the scheme, an FHE should meet two additional requirements, *compactness* and *circuit privacy* [5]. Achieving compactness poses a great challenge, which means the size of the ciphertext should remain same (within the required bounds) irrespective of the function being evaluated. In his first construction, Gentry has formulated a three-step procedure to obtain an FHE scheme satisfying these requirements. This procedure includes, 1) Constructing a *Somewhat Homomorphic Encryption* (SHE) scheme that supports many additions, but, only a limited number of multiplications, 2) *Squashing* the decryption function of the SHE so that the scheme can evaluate its own decryption function (bootstrappability), and 3) obtaining the FHE by applying a *ciphertext refreshing* method periodically, for *bootstrapping*, to bring back the *noise* in the ciphertexts to the required low level once it exceeds a threshold value [2][3][6].

The FHE schemes developed so far based on the above blueprint are inefficient and impractical because of the colossal difference between the computational complexities of processing the ciphertexts and the corresponding plaintexts [12]. The major factors contributing to these high computational complexities are *huge public key*, *large message expansion* and the ciphertext refreshing *Recrypt* procedure.

Many of the existing homomorphic encryption schemes support unlimited additions, and supporting unlimited multiplications is the main hindrance. In fact, the ciphertext refreshing procedure in the Gentry's FHE is introduced to allow for unlimited multiplications on ciphertexts. The issue of practicality of the FHE schemes arise several important questions. Principally, is it really necessary to follow the above blueprint for constructing an FHE scheme? How many multiplications on ciphertexts are required for any application in practice for supporting encrypted data processing? Or in other words, do we really need an FHE with capability of supporting unlimited multiplications? Several works [17][31][32] have tried to provide answers to these questions in the form of developing an SHE scheme suitable for certain practical applications. The essence is that, in practice there are several applications which involve many additions but, a few number of multiplications in the functions they use for manipulation of data, and hence an SHE scheme is sufficient for processing the encrypted data in these applications [11][17][19][20]. Despite this fact, no practical SHE schemes exist yet.

A. Related work and recent advances

Soon after the Gentry's FHE invention, three major variant schemes have appeared following the blueprint of his

DOI: 01.IJNS.3.4.1071

construction. The first of these was devised by Smart and Vercauteren [4], the second one was by Van Dijk et.al. [5], and the third variant was by Brakerski and Vaikuntanathan [11]. Stehlé and Steinfeld [7] suggested two optimizations to Gentry's scheme that lead to improvement in the complexity of decryption process from $\tilde{O}(n^6)$ to $\tilde{O}(n^{3.5})$. Ogura et al. [8], Scholl and Smart [15] have proposed improvements to the key generation algorithm of Gentry's FHE scheme. The FHE of Brakerski and Vaikuntanathan [10] eliminated the step-2, i.e., squashing the decryption function, of Gentry's blueprint. A different technique was used by Gentry and Halevi [16] to eliminate this squashing step, which involves expressing the decryption function of the SHE as a depth-3 arithmetic circuit and switching between Multiplicatively Homomorphic Encryption (MHE) mode and SHE mode during the homomorphic evaluation of that circuit. Lauter, Naehrig and Vaikuntanathan [17] demonstrated the construction of an SHE, which can efficiently evaluate low degree functions. Brakerski et al.'s work [12] completely eliminated the bootstrapping process.

The first attempt in the implementation of an FHE is by Smart and Vercauteren [4], but, they could not implement the bootstrappable version due to the assumption that the determinant of the lattice they used should be prime. Eliminating this prime determinant requirement and combining with several optimizations Gentry and Halevi [18] demonstrated the first implementation of the Gentry's original ideal lattice based scheme. Coron et.al.[9] have described the first implementation of integer based FHE scheme of [5]. Their major contribution was in reducing the public key size of the scheme in [5] from $\tilde{O}(n^{10})$ to $\tilde{O}(n^7)$. Other efforts in implementing the variants of Gentry's scheme are, integer based symmetric key FHE implementation by Jibang Liu et al. [21], proof-of-concept implementation of Brakerski et al.'s SHE scheme [11] by Lauter et al. [17], Gentry's SHE implementation by Michal MikuŠ [13] and Smart and Vercauteren's FHE implementation by Henning Perl et al. [14]. The work of Vinod Vaikuntanathan [19] provides an expository survey of the recent advances in homomorphic cryptography. Jing-Li et al. [28] described the extension of Gentry's scheme [3] to larger message space. Govinda Ramaiah and Vijaya Kumari [29] and Hao-Miao Yang et al. [31] have proposed similar variant of the integers based scheme of [5] in separate works with an efficient public key generation method, which leads to a public key of size $\tilde{O}(n^3)$. Nevertheless, the scheme proposed by [29] uses a simple and straightforward method to achieve compactness. Coron et al.'s work [30] was an optimization of their previous work [9] in reducing the size of the public key of [5] to $\tilde{O}(n^5)$. Very recently another variant SHE is proposed by Govinda Ramaiah and Vijaya Kumari [32]. This scheme is capable of encrypting many bits together or integer plaintexts.

B. Contributions of this work

This paper presents a more concrete and secure version of the SHE theoretically proposed in [29] with implementation and more tangible performance details. Experimental results show that, the computational complexities are drastically reduced, when compared to any other integer based SHE scheme existing today. This makes the proposed solution close to deployment in suitable practical applications. The method of key generation when combined with ciphertext refreshing procedure described in [5] using the optimization suggested by [7] leads to an efficient FHE scheme comparatively.

II. Preliminaries

A. Notation and basics

In this paper, lower case italic letters denote the parameters used to represent sizes (bit-length) of various integers. Similarly, upper case letters denote the integers and real numbers, and bold upper case letters denote the sets. $\lfloor X \rceil$ indicates rounding of the real number X *to the nearest integer* that is unique in the open interval (X-1/2, X+1/2]. The quotient and remainder resulting from the division Z/P are designated by $Q_P(Z) = \lfloor Z/P \rceil$, and $R_P(Z) = Z - Q_P(Z)P$, respectively . The notation $[Z]_P$ or Z mod P is used interchangeably to represent modulo operation of Z with respect to P, which results in $R_P(Z)$. Since $Q_P(Z)$ is defined by rounding to the nearest integer, $R_P(Z) \in (-P/2, P/2]$ when P is odd. lg X designates the logarithm of X to the base 2. Choosing a random integer X uniformly from a finite set **S** is indicated as $X \xleftarrow{\$} \mathbf{S}$. The soft-oh notation $f(n) = \tilde{O}(g(n))$ is used to represent $f(n) = O(g(n)\lg^k g(n))$ for some k, ignoring the logarithmic factors and any other smaller additive complexities. A K-rough integer is an integer not having prime factors smaller than the integer K. It is suggested to refer [2], [3] and [5] for various definitions related to Fully Homomorphic Encryption.

B. The DGHV scheme

In this section, the construction of Van Dijk et.al's SHE scheme over the integers [5] is described. Let, *n* denote the security parameter.

e denotes size of the secret key integer. In order to support homomorphism for sufficiently deeper circuits, *e* is taken as $\geq r\,\Theta(n \lg^2 n)$.

The public key consists of many approximate multiples of the secret key integer. The approximate multiple of an integer is obtained by adding a small error or *noise* integer to its exact multiple.

t denotes the number of integers in the public key. To use the leftover hash lemma (Lemma 2.1, [5]) in reducing the security of the scheme to solving AGCD problem (defined below), *t* is taken as $\geq g + \omega(\lg n)$

r denotes size of the noise in each of the public key integers. To foil the brute-force attack against the noise, *r* is taken as $\omega(\lg n)$.

g denotes the size of each public key integer. For security against the lattice based attacks on the underlying AGCD problem, *g* is taken as $\omega(e^2 \lg n)$.

DOI: 01.IJNS.3.4.1071

d denotes the size of the additional noise ($d > r$) used during the encryption of a plaintext bit.

The parameter setting suggested by [5], claiming a complexity of $\tilde{O}(n^{10})$ is,

$e = \tilde{O}(n^2)$, $r = n$, $d = 2n$, $g = \tilde{O}(n^5)$, $t = g + n$

Figure.1 shows the DGHV's SHE scheme with respect to the above parameters.

KeyGen (n):

1. Sample $P \xleftarrow{\$} (2\mathbb{Z}+1) \cap [2^{e-1}, 2^e)$ to generate the secret key. i.e., secret key is a random e-bit odd integer P.
2. For i = 0, 1, …., t sample $X_i \xleftarrow{\$} D_{g,r}(P)$, where the distribution, $D_{g,r}(P)$ $= \{Q \xleftarrow{\$} \mathbb{Z} \cap [0, 2^g/P), R \xleftarrow{\$} \mathbb{Z} \cap (-2^r, 2^r) : X = PQ + R\}$ until the following conditions are satisfied. $X_0 > X_1, \ldots, X_t$, $[X_0]_2 = 1$, and $[[X_0]_P]_2 = 0$
3. Output the public key $PK = (X_0, X_1, \ldots, X_t)$ and the secret key SK = P.

Encrypt (PK, $M \in \{0, 1\}$):

1. Choose noise for encryption: $B \xleftarrow{\$} \mathbb{Z} \cap (-2^d, 2^d)$.
2. Choose a subset $J \subseteq \{1, \ldots, t\}$.
3. Compute the sum $S = \sum_{i \in J} X_i$
4. Output the ciphertext, $C = [M + 2(B + S)]_{X_0}$

Decrypt (SK, C): Compute $M = [[C]_P]_2$

Evaluate (PK, CKT, $(C_1, \ldots, C_k)$): Let CKT be the binary circuit to be evaluated representing a Boolean function f, with XOR gates and AND gates (i.e., CKT consists of mod-2 addition and multiplication gates). Replace the XOR gates and AND gates of C with addition and multiplication gates that operate over integers. Let CKT_g be the resulting generalized circuit and f_g be the corresponding multivariate polynomial. Apply CKT_g over the ciphertext integers $(C_1, \ldots, C_k)$, and output the resulting ciphertext C_g that corresponds to $f_g(C_1, \ldots, C_k)$.

Figure1. The DGHV Somewhat Homomorphic Encryption Scheme

The size of the public key in this scheme is $\tilde{O}(n^{10})$ because, the public key consists of $t = \tilde{O}(n^5)$ integers each of size $g = \tilde{O}(n^5)$. When the ciphertext expression (step 4 in *Encrypt*) is expanded, it takes the form (M+2B+PQ). The term (M+2B) << P, is the noise (the distance to the multiple PQ), which makes C an approximate or near multiple of P. The main problem which makes the scheme Somewhat Homomorphic is the rapid growth in this noise during multiplication operation in *Evaluate*. For every multiplication, the bit length of the resulting noise equals the sum of the bit lengths of the multiplicand noises, which crosses the size of P/2 after certain number of multiplications, resulting in incorrect decryption. Further, since P is odd it will not influence the parity of $\lfloor C/P \rceil$, and thus the decryption function can be written as $[[C]_p]_2 = [C - \lfloor C/P \rceil]_2$, which is equal to the XOR of the Least Significant Bits (LSBs) of C and $\lfloor C/P \rceil$, i.e., $[[C]_p]_2 = LSB(C) \oplus LSB(\lfloor C/P \rceil)$. Even though this *squashed* version of the decryption function that involves a single gate applied to only two bits looks simple, the computation of $\lfloor C/P \rceil$ is so complex that the decryption circuit cannot handle it [6]. To make the scheme *bootstrappable* and consequently obtaining FHE by overcoming this problem, the *ciphertext refreshing* procedure suggested in Gentry's blue print is applied. Since the optimizations suggested in this work target only the underlying SHE, discussion is restricted to the same in this paper. DGHV suggested different optimizations for achieving compactness. The simplest of those optimizations involve publishing an exact multiple of the secret key P, and reducing the ciphertext modulo that exact multiple after every addition and multiplication in *Evaluate*. This method is followed by [9] also, and the same technique is used for compactness of ciphertexts in the proposed scheme. The security of DGHV scheme is based on the hard problem of solving Approximate Greatest Common Divisors (AGCD), which can be defined as follows.

Approximate Greatest Common Divisors Problem: The (*r*, *e*, g)–Approximate Greatest Common Divisors problem is, given polynomially many samples from the distribution $D_{g,r}(P)$, for a randomly choosen *e*-bit odd integer P, output P.

III. The GV scheme

In this section a more concrete and secure version of the variant scheme proposed by Govinda Ramaiah and Vijaya Kumari [29] with an efficient public key generation method is presented. The public key consists of two big integers X_0 and X_1. Integer X_0 is an exact multiple of the odd secret integer P and X_1 is an approximate multiple, i.e., multiple of P containing some additive error R. To encrypt a plaintext bit M, first the erroneous integer X_1 of the public key is added with some more additional noise R', resulting in another big integer say X_2. This X_2 is now multiplied with a random even integer 2N, the result is added to the plaintext bit and the final sum is reduced modulo the error-free integer X_0 in the public key. For homomorphic evaluation of a function, the addition and multiplication operations in the corresponding generalized binary circuit are performed over ciphertexts by reducing the result of each addition and multiplication modulo the error-free integer X_0 in the public key. The security of the scheme is reduced to the two-element Partial Approximate Greatest Common Divisors (PAGCD) problem. The parameter setting for the GV variant scheme is reviewed as follows.

For the given security parameter *n*,

e denotes the size of the secret key integer P. For achieving homomorphism in evaluation of sufficiently deeper circuits, *e* is taken as $\geq d\ \Theta(n \lg^2 n)$.

d is size of the multiplicative noise integer used for encryption. To avoid the brute-force attack against it, the size of this integer is taken as $\geq 2n$.

r is the size of the noise in the public key integer X_1, which is taken as $\omega(\lg n)$ to foil the brute-force attack against the noise.

g is the number of bits in each of the public key integers. Roughly, *g* is the size of the factor Q in the multiples of P, in the public key. Since the public key consists of only two elements, the attacks related to two-element PAGCD problem (section IV) only are considered. Hence it is claimed that, it is sufficient to take $g > e$ against the condition used in [5] as $g > e^2$ to thwart lattice based attacks on the AGCD problem with some arbitrary *t* number

DOI: 01.IJNS.3.4.1071

of elements. Therefore, g is taken as $\omega(e \lg n)$.

The suggested parameter setting with respect to the above discussion is, $e = \tilde{O}(n^2)$, $r = n$, $d = 2n$, and $g = \tilde{O}(n^3)$. The GV's SHE scheme is shown in Figure2. Superscript SP denotes that the algorithms are related to the proposed variant with **S**maller **P**ublic key.

KeyGen$^{SP}(n)$:

1. Sample $P \xleftarrow{\$} (2\mathbb{Z}+1)) \cap [2^{e-1}, 2^e)$ to generate the secret key. i.e., secret key is a random e-bit odd integer P.
2. Sample $R \xleftarrow{\$} \mathbb{Z} \cap (-2^r, 2^r)$ to generate the noise for the public key integer.
3. Choose two g-bit random integers Q_0, Q_1. For this, sample $Q_i \xleftarrow{\$} \mathbb{Z} \cap [0, 2^g / P)$, for i = 0,1.
4. Compute $X_0 = PQ_0$, and $X_1 = PQ_1 + R$.
5. Repeat the steps 2 – 4, if $[X_0]_2 = 0$. That means, X_0 will be an odd integer. The condition $X_0 > X_1$ specified in the DGHV scheme is omitted.
6. Output the secret key SK = P and the public key PK = (X_0, X_1).

Encrypt^{SP} (PK, $M \in \{0,1\}$): To encrypt a plaintext bit $M \in \{0,1\}$,

1. Choose a random d-bit integer N. For this, sample $N \xleftarrow{\$} \mathbb{Z} \cap [2^{d-1}, 2^d)$.
2. Sample $R' \xleftarrow{\$} \mathbb{Z} \cap (-2^r, 2^r)$ to generate additional noise for the public key integer X_1.
3. Compute $X_2 = X_1 + R'$
4. Output the ciphertext $C = [M + 2NX_2]_{X_0}$

Decrypt^{SP}(SK, C) and *Evaluate*^{SP}(PK, CKT, $(C_1,\ldots,C_k)$) algorithms are same as that of the DGHV scheme with the only difference that, for any two ciphertexts C_1 and C_2 during the generalized circuit evaluation, every addition and multiplication operation is performed as,

Add : Compute $C_a = [C_1 + C_2] \bmod X_0$, and
Mul : Compute $C_m = [C_1 C_2] \bmod X_0$

Figure2. The GV Somewhat Homomorphic Encryption Scheme

The appealing feature of this scheme is the smaller public key with only two integers of size $\tilde{O}(n^3)$ each. It is quite easy to see that GV scheme is a *variant of the* DGHV *scheme for the chosen parameter setting*. For *Evaluate*^{SP}, corresponding to the generalized circuit CKT_g we have the following notion of permitted circuit.

Permitted circuit: An arithmetic circuit with addition and multiplication gates is called a permitted circuit for the GV scheme if, for any set of integer inputs each $< 2^d$ in absolute value, the maximum absolute value output by the circuit is $< 2^{e-2}$. We denote the set of permitted circuits as $\mathbf{CKT_p}$.

Theorem1. The GV scheme proposed is correct, compact and is algebraically homomorphic for the given plaintext $M \in \{0,1\}$, and for any circuit $CKT \in \mathbf{CKT_p}$.

Proof: Let us consider the fresh ciphertext output by *Encrypt*^{SP}(PK, M). We have, $C = [M + 2NX_2] \bmod X_0$
$= [M + 2N(R'+R+PQ_1)] \bmod X_0$
$= M+2N(R'+R)+P(2NQ_1 - KQ_0)$ for some integer $K \geq 0$
$= M+2B+PQ'$ where $B=N(R'+R)$, and $Q'=(2NQ_1 - KQ_0)$. For correct decryption of a ciphertext, the absolute value of the term (M+2B) should be always less than P/2. For the fresh ciphertexts, it is enough to verify the sizes of these values. For the chosen parameters we have the bit length of (M+2B) is $d = 2n$ and the bit length of P is $e = n^2$. This shows that, the condition for correct decryption just discussed is satisfied. Hence, *Decrypt*^{SP} works properly for all the fresh ciphertexts. To prove the correctness with respect to *Evaluate*^{SP}, we use the notion of permitted circuit. It should be noted that, the integers mentioned in the definition of the permitted circuit corresponds to the noise in the ciphertexts. The noise in the fresh ciphertexts will be $< 2^d$ and we have $2^{e-2} = P/2$, for some odd integer $P \in [2^{e-1}, 2^e)$. That means, when a permitted circuit is applied to a set of ciphertext integers, the maximum absolute value of the noise should be less than P/2, where P is the minimum value of the secret key integer that can be chosen. Now, in *Evaluate*^{SP}, let the circuit CKT_g is applied to the ciphertext integers $C_1, \ldots, C_k$ and the resulting ciphertext is C. Since the circuit $CKT_g \in \mathbf{CKT_p}$, the resulting noise in C after the *Evaluate*^{SP} will be less than P/2 for any P chosen from the specified interval for that particular instance of the scheme, which shows that the decryption works properly proving the correctness of *Evaluate*^{SP} .

It is evident that the modular reduction results in an integer the size of which is always less than or equal to the modulus. Therefore, reduction with X_0 produces an integer with size $\leq |X_0|$. Since, $|X_0|$ is $\tilde{O}(n^3)$, this defines the bound for the ciphertext compactness. Thus, the size of the ciphertext resulting from *Evaluate*^{SP} is always $\leq \tilde{O}(n^3)$ irrespective of the circuit CKT being evaluated, which proves the compactness of the scheme. It can be easily verified that the modular reduction with X_0 affects only the PQ term in a ciphertext and the noise remains unaltered. Hence, decryption works properly even after the modular reduction with X_0.

Let, $C_1=(M_1+2B_1+PQ_1')$, and $C_2=(M_2+2B_2+PQ_2')$ Addition in *Evaluate*^{SP} gives,
$C_a=C_1+C_2 = (M_1+M_2)+2B_a+PQ_a$, for some integers B_a and Q_a. Similarly, multiplication in *Evaluate*^{SP} gives,
$C_m = C_1C_2 = (M_1M_2)+2B_m+PQ_m$, for some integers B_m and Q_m. It can be seen that for the given values of R and P, the values corresponding to the noise B and the integer Q in the fresh ciphertexts as well as the ciphertexts resulting from *Evaluate*^{SP}, all belong to the same intervals $[2^{d-1}, 2^d)$ and $[0, 2^g/P)$ respectively whenever the circuit being evaluated $CKT \in \mathbf{CKT_p}$. Hence, C_a decrypts to (M_1+M_2) and C_m decrypts to (M_1M_2) correctly. □

Lemma 1. *Let $f(x_0,\ldots,x_k)$ be a multivariate polynomial in k variables with degree m and* CKT *be the corresponding arithmetic circuit. Then,* $CKT \in \mathbf{CKT_p}$ if $|\overline{f}|(2^d)^m \leq 2^{e-2}$, *where $|\overline{f}|$ is the l_1* norm *of the coefficient vector of f.* □

The above lemma defines the multiplicative capacity of the scheme and in turn the set of *permitted polynomials*. The number of multiplications supported corresponds to the degree m of the permitted polynomials, which can be given as,

$$m \leq (e - 2 - \lg |\overline{f}|) / d$$

DOI: 01.IJNS.3.4.1071

IV. Security of the GV scheme

Since GV scheme is a variant of the DGHV's SHE, the same strategy employed by [5] and [9] can be followed to base the security of the scheme on the hard problem of solving a version of AGCD called Partial Approximate Greatest Common Divisor (PAGCD).

Two-element PAGCD problem: The two-element (*r, e, g*) - PAGCD problem is: For a random *e*-bit odd positive integer P, given $X_0 = PQ_0$ and $X_1 = PQ_1 + R$, where Q_i (i=0,1), R are chosen from the intervals $[0, 2^g/P)$, and $(-2^r, 2^r)$ respectively, output P.

Chen and Nguyen [23] shown that, solving PAGCD is relatively easier than solving General AGCD. However, as mentioned by them their attack's implementation parameters are suboptimal for medium and large challenges put forth by Coron et. al.[9]. Hence, if the security parameter *n* is appropriately chosen, the PAGCD problem will be intractable ensuring the semantic security of the scheme. We have the following theorem, similar to [5] in order to reduce the security of our scheme to the two-element PAGCD problem.

Theorem 2. *Let e, g, r, d be the parameters of the GV scheme, which are polynomial in the security parameter n. An adversary $\mathcal{A}$ with an advantage ε against the scheme can be converted in to an algorithm B for solving the two-element (r, e, g)-PAGCD problem with success probability at least /2. The running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, n and 1/*

Proof: Let $\mathcal{A}$ be the adversary against the GV scheme. The input to A is the public key generated by *KeyGen*^{SP} and the ciphertext produced by *Encrypt*^{SP}. The output of A is the plaintext bit M with probability $1/2+\varepsilon$ for a non-negligible . Now, as described in [5], the algorithm B that solves the two element (*r, e, g*)-PAGCD problem is as follows.

To start with, $\mathcal{B}$ takes the two public key integers $X_0 = PQ_0$, $X_1 = PQ_1 + R$ corresponding to an odd *e*-bit secret key integer P, as generated by *KeyGen*^{SP}. Using the subroutine given in Figure3, $\mathcal{B}$ finds the LSB of the quotient $Q_p(Z)$ of a given integer $Z \in [0, 2^g)$, with $|R_p(Z)| < 2^r$.

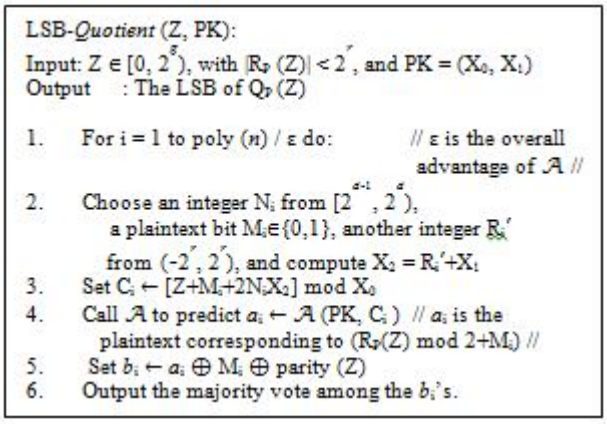

Figure 3.Subroutine for predicting LSB of the quotient

In the line 5 of LSB-*Quotient* b_i is the parity of $Q_p(Z)$. This is because, since P is odd we have, Parity($Q_p(Z)$)= Parity($R_p(Z)$) $\oplus$ Parity(Z). In the following Lemma 2 we show that, the ciphertext C_i in line 3 of LSB-*Quotient* is distributed almost identically to a valid encryption of $([R_p(Z)]_2+M_i)$ for all but a negligible fraction of the public keys generated by the GV scheme. Therefore, $\mathcal{A}$ has a noticeable advantage in guessing the plaintext bit encrypted under PK. Due to this, LSB-*Quotient* will output LSB of ($Q_p(Z)$) with overwhelming probability.

For any two integers $Z_1 = R_p(Z_1)+PQ_p(Z_1)$ and $Z_2 = R_p(Z_2)+PQ_p(Z_2)$, where the difference in the sizes of $R_p(Z_i)$ and P are as mentioned in the parameter setting, apply the binary-GCD algorithm as described in [5] finding the parity of both $Q_p(Z_1)$ and $Q_p(Z_2)$ from LSB-*Quotient* above. After about O(*g*) iterations we get two integers say Z_1', Z_2' with $Q_p(Z_2') = 0$ and $Q_p(Z_1') = 1$, which is the odd part of $GCD(Q_p(Z_1), Q_p(Z_2))$. That means, we get finally some integer Z =binary-$GCD(Z_1,Z_2) = GCD(Q_p(Z_1), Q_p(Z_2))P+R_n = 1P+R_n$ for some noise R_n.

To Recover P, the algorithm $\mathcal{B}$ takes the two public key integers X_0, X_1 and applies the binary-GCD algorithm to them. Note that X_0, X_1 are generated so that they have a common hidden P. Also, since Q_0, Q_1 (i.e., $Q_p(X_0)$, $Q_p(X_1)$) are chosen at random from a very large interval, according to the theorem D, page 342, of [27], there is a very good probability of $6/\pi^2 \cong 0.61$, that Q_0, Q_1 are co-prime (i.e., odd part of $GCD(Q_0, Q_1)$ is 1). Hence, $GCD(X_0, X_1)$ will return some $Z = 1P + R_n$, with $|R_n| < 2^r$ with high enough probability. Now, again when the binary-GCD is applied to (X_0, Z), the sequence of parity bits of $Q_p(X_0)$ in all the iterations will be the binary representation of $Q_p(X_0) = Q_0$. Since X_0 is an exact multiple of P, $P = X_0 / Q_0$.

Thus, $\mathcal{B}$ is a solver of PAGCD and its overall success probability as analyzed in [5] is $\varepsilon/2$ and its running time is polynomial in the running time of $\mathcal{A}$, n and $1/\varepsilon$. □

Lemma 2. *Let the parameters e, g, r, d are set as specified in the scheme. The secret key* SK = P *and the public key* PK = (X_0, X_1) *are chosen according to the KeyGen*^{SP}. *For every integer* $Z \in [0, 2^g)$, *which is* 2^r*away from a multiple of* P, *consider the following distribution.*

$C_{PK}(Z)$={ $N \xleftarrow{\$} \mathbb{Z} \cap [2^{d-1}, 2^d)$, $R' \xleftarrow{\$} \mathbb{Z} \cap (-2^r, 2^r)$: *Output* $C' = [Z + 2N(R'+X_1)]_{X_0}$ }

$C_{PK}(Z)$ *is same as the distribution induced over the ciphertexts generated by Encrypt*^{SP}(PK, M = $[[Z]_p]_2$), *with overwhelming probability.*

Proof: For a given ciphertext C, the integer $Q_p(C)$ is uniform in $(-Q_0/2, Q_0/2]$ from the leftover hash lemma (Lemma 2.1, [5]). Let $C'=[Z+2N(R'+X_1)]_{X_0} = M'+2B'+PQ'$, for some integers B' and Q'. Thus, the integer Z used to generate C' can be treated as a ciphertext generated by *Encrypt*^{SP}(PK, M'). This is because, Z is 2^raway from a multiple of P. Therefore, C' corresponds to the result of homomorphic addition of two ciphertexts corresponding to the plaintext bit (M' + M) with M = 0. Now, Theorem1 shows that C' produces correct plaintext up on decryption. Also, consider the integer hash function family, h': $[2^{d-1}, 2^d) \longrightarrow \mathbb{Z}_{Q_0}$, where $\mathbb{Z}_{Q_0}$ denotes integers mod Q_0 , and $h'(N) = [2NQ_1]_{Q_0}$, with N chosen from $[2^{d-1}, 2^d)$. Clearly h' is pair wise independent because, for any two integers N_1, N_2 chosen from $[2^{d-1}, 2^d)$, with $N_1 \neq N_2$,

DOI: 01.IJNS.3.4.1071

$Pr_h[h'(N_1)= h'(N_2)]$ is quite negligible and is almost zero. This is because, $|Q_0| >> |N|$ and since Q_1, Q_0 are fixed, modular reduction of $2N(R'+X_1)$ with Q_0 always produces different remainders for different values of N. Hence in C', the value of Q' is also uniform in $(-Q_0/2, Q_0/2]$ and the claim follows. □

V. Known attacks

In the GV scheme, for a given security parameter *n* the lowest possible size of the problem to solve the PAGCD problem is the public key (X_0, X_1) for a given secret key integer P because, the noise in X_1 is very much less when compared to the noise in the ciphertexts for a particular instance of the scheme. Therefore, the attacks against the two-element PAGCD problem only are described, i.e., against the public key only, claiming that the high noise ciphertexts (approximate multiples of P) successfully defend all these attacks.

1) Factoring the exact multiple: For the chosen parameter values, the size of the exact multiple of P i.e., X_0 is big enough so that, even the best known integer factoring algorithms such as the General Number Field Sieve [24] will not be able to factor X_0. Even if the factor P is targeted which is smaller than the size of total Q_0, algorithms such as Lenstra's elliptic curve factoring [25] takes about $\exp(O(\sqrt{e}))$time to find P. But, it is to be noted that P will not be recovered directly as it is not prime and may be further decomposed in to smaller primes. For enhanced security, X_0 may be generated with P and Q_0 as 2^{1024}-rough integers as discussed in [32].

2) Brute-force attack on the noise: Given the public key integers $X_0 = PQ_0$ and $X_1 = PQ_1 + R$, where size of R is $\tilde{O}(n)$, the simple brute-force attack is: choosing an R form the interval $(-2^r, 2^r)$, subtracting it from X_1, and computing $GCD(X_0, X_1 - R)$ every time, which may be the required secret integer P. In a worst case, this process may need to be repeated for all the integers R in the specified interval. The complexity of this attack would be $2^r \tilde{O}(g)$ for *g*-bit integers.

3) Continued fractions and lattice based attacks: Howgrave Graham [22] described two methods to solve the two-element PAGCD problem. In simple terms the continued fraction based approach (Algorithm 11, [22]) recovers P if the condition $R < P/Q$ is satisfied. Similarly, his lattice based algorithm (Algorithm 12, [22]) recovers P if the condition $R < P^2 / (PQ)^\varepsilon$ is satisfied for some real number ε in (0,..,1). Also, for the case of a two-element PAGCD problem, it is possible to recover P when r/g is smaller than $(e/g)^2$ [5]. Since the chosen parameter setting does not satisfy these constraints the concerned methods fail to recover the value of P.

VI. Performance and practicality

A. Improvement in bit complexity

The public key size of the DGHV scheme is $\tilde{O}(n^{10})$. Generation of each public key element involves $\tilde{O}(n^5 . n^2)$ bit operations. This will take $\tilde{O}(n^{12})$ computations to generate complete public key, which contains $\tilde{O}(n^5)$ elements. The ciphertext expansion in that scheme is n^5.

The public key in the GV variant consists of only two elements each having a size of $\tilde{O}(n^3)$ bits. Hence, the size of the public key is $\tilde{O}(n^3)$. The key generation complexity is $\tilde{O}(n^3.n^2) = \tilde{O}(n^5)$ for generating two public key elements. This is a significant improvement over the SHE schemes of [5] and [9]. The encryption of a single bit plaintext, which involves a multiplication of $\tilde{O}(n^3 . n)$ and a modular reduction of the resulting $\tilde{O}(n^3)$-bit integer with $\tilde{O}(n^3)$-bit X_0 takes $\tilde{O}(n^6)$ steps. The major factor contributing to the bit complexity of decryption is the modular reduction of $\tilde{O}(n^3)$-bit ciphertext with the $\tilde{O}(n^2)$-bit secret key integer P. This makes the decryption complexity roughly $\tilde{O}(n^6)$. Therefore, the overall theoretical complexity of GV variant is $\tilde{O}(n^6)$. Since a single bit plaintext is expanded to a ciphertext of $\tilde{O}(n^3)$ bits, the expansion ratio is also less, which is n^3. TABLE I summarizes the comparative analysis with existing integer based SHE schemes.

B. Experimental results

The GV scheme is implemented in Visual C++ 2008 Express edition using Victor Shoup's Number Theory Library (NTL) [33] for the manipulation of big integers involved. The programs were run on a normal desktop PC with Intel Core 2 Duo T5750 2GHz processor and 4GB RAM, in Windows 7 Professional operating system environment. Experimentation was carried out to measure the time taken for various algorithms in the scheme, with different values of the security parameter.

Table I. Summary Of Improvements

Item of comparison	DGHV SHE [5]	CMNT SHE [9]	GV SHE
Compactness	-No-	Yes	Yes
Size of the public key	$\tilde{O}(n^{10})$	$\tilde{O}(n^7)$	$\tilde{O}(n^3)$
KeyGen complexity	$\tilde{O}(n^{12})$	$\tilde{O}(n^9)$	$\tilde{O}(n^5)$
Encrypt complexity	$\tilde{O}(n^{10})$	$\tilde{O}(n^{15})$	$\tilde{O}(n^6)$
Decrypt complexity	$\tilde{O}(n^{10})$	$\tilde{O}(n^{10})$	$\tilde{O}(n^6)$
Message expansion	n^5	n^5	n^3
Overall complexity	$\tilde{O}(n^{12})$	$\tilde{O}(n^{15})$	$\tilde{O}(n^6)$
Security base	AGCD	PAGCD (Error-free approximate GCD)	Two-element PAGCD

The practical multiplicative capacity of the scheme is obtained at various levels of security. TABLE II below shows the values of parameters corresponding to different security levels: Toy, Small, Medium, and Large. The results correspond to the encryption of a single bit. All the times in TABLE III and TABLE IV are shown in seconds.

Table II. Values Of Parameters At Different Security Levels

Level of Security	*n*	*e*	*r*	*d*	*g*
Toy	32	1024	32	64	32768
Small	64	4096	64	128	262144
Medium	80	6400	80	160	512000
Large	128	16384	128	256	2097152

TABLE V gives the comparision of practical performances (time in seconds) of the GV scheme with two other implementations [9][30], corresponding to the value of the

DOI: 01.IJNS.3.4.1071

TABLE III. PRACTICAL PERFORMANCE AT DIFFERENT SECURITY LEVELS

Level of Security	Security Parameter n	KeyGen Time	Encrypt Time	Decrypt Time
Toy	32	0.015	0	0
Small	64	0.047	0	0.015
Medium	80	0.187	0.016	0.062
Large	128	0.607	0.031	0.686

TABLE IV. PRACTICAL EVALUATION CAPACITY OF THE SCHEME

n	Time for Addition of two bits	Time for Multiplication of two bits	Number of multiplications over a fresh ciphertext
32	0	0.047	10
64	0	2.402	20
80	0	8.986	25
128	0	94	40

security parameter $n = 72$.

Analysis of experimental results show that, the GV scheme is quite efficient than any other integer based SHE scheme existing. This drastic improvement in performance makes the GV scheme ready for deployment in suitable practical applications. However, the evaluation results correspond to operations over the ciphertexts that encrypt single bits. Thus, the impracticality of the scheme now can be totally attributed to the size of the input function or circuit to the $Evaluate^{SP}$ algorithm. Hence, the reduced complexities combined with the ability to encrypt many bits at once or integer plaintexts as done in [32] makes the scheme really practical.

TABLE V. COMPARISON WITH EXISTING IMPLEMENTATIONS

Algorithm for $n = 72$	CMNT Scheme [9]	CNT Scheme [30]	GV Scheme (Current work)
KeyGen	2580	378	0.062
Encrypt	177	3.4	0
Decrypt	0.05	0	0.047
Evaluate (Multiplication)	Not available	41	4.805

C. *Applications*

The GV scheme is suitable for all the applications which involve the functions that contain many additions, but, few multiplications. For example, as shown in Table IV the scheme supports nearly 40 multiplications for large instance.

Lauter et al.[17], Brakerski and Vaikuntanathan [10][11][19] and [26] have discussed the applications for which an SHE scheme is quite sufficient for encrypted data processing and allow the delegation of computation to a cloud server. The efficient GV scheme can be practically implemented in all such applications. There are two categories of applications for a homomorphic encryption scheme in practice [17]. 1) Applications that demand encryption of both data and functions to be computed, e.g. *Cloud based financial information systems* and 2) Applications that need only encryption of data, e.g. *Cloud based healthcare services*. Such applications involve simple statistical functions like average, standard deviation, and logistical regression. Evaluation of these functions requires many additions and one or few multiplications. However, due to lack of proper methods for computing square roots and divisions on real numbers homomorphically, computation of such operations should be done after decrypting the encrypted sums and products.

Another application area that is closely related to the homomorphic cryptography is *Private Information Retrieval* (PIR) [19] [10] [20]. In the PIR protocol, a large database is maintained by the cloud server and the customer likes to retrieve a particular entry from that database privately. Customer sends the encrypted index that is to be queried. The cloud server homomorphically evaluates the database access function to retrieve the required entry in the database in encrypted form using the encrypted index, and sends the result to the customer. Bitwise encryption of index is a drawback in such situations which leads to high communication complexity, and as a solution [10] proposes the use of a symmetric key encryption in combination with public key homomorphic scheme.

CONCLUSIONS

In this paper, an efficient variant of the DGHV's SHE scheme is presented with experimentation details. The security of the GV scheme proposed is based on the hard problem of solving the two-element PAGCD. Due to the smaller public key that contains two integers of size $\tilde{O}(n^3)$ each, the overall complexity is halved with reduction from $\tilde{O}(n^{12})$ of the DGHV scheme to $\tilde{O}(n^6)$ in the GV variant. Experimental results prove that the performance of the proposed SHE is very close to the practicality. The applications for which the scheme is suitable for practical deployment are discussed.

REFERENCES

[1] R. Rivest, L. Adleman, M. Dertouzos, M "On data banks and privacy homomorphisms", Foundations of Secure Computation, pp. 169–180, 1978.

[2] C. Gentry, "A fully Homomorphic Encryption scheme", Ph.D Thesis, Stanford University, 2009

[3] C. Gentry, "Fully Homomorphic Encryption using ideal lattices", In STOC, pp 169-178, ACM, 2009.

[4] N. P. Smart, F. Vercauteren, "Fully Homomorphic Encryption with relatively small key and ciphertext sizes" In Public Key Cryptography - PKC'10, Vol. 6056 of LNCS, pp. 420-443,Springer, 2010

[5] M. V. Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, "Fully homomorphic encryption over the integers", Proceedings of Eurocrypt, Vol. 6110 of LNCS, pp. 24-43, Springer, 2010.

[6] C. Gentry, "Computing arbitrary functions of encrypted data", Communications of the ACM, 53(3), pp.97-105, 2010.

[7] D. Stehlé, R. Steinfeld, "Faster fully homomorphic encryption. ASIACRYPT'2010, Vol. 6477 of LNCS, pp.377-394, Springer, 2010

[8] N. Ogura, G. Yamamoto, T. Kobayashi, S. Uchiyama, "An improvement of key generation algorithm for Gentry's homomorphic encryption scheme", Advances in Information and Computer Security - IWSEC 2010, Vol. 6434 of LNCS, pp. 70–83, Springer, 2010.

[9] J. S. Coron, A. Mandal, D. Naccache and M. Tibouchi, "Fully Homomorphic Encryption over the Integers with Shorter

DOI: 01.IJNS.3.4.1071

Public Keys", CRYPTO 2011, P. Rogaway (Ed.), Vol. 6841 of LNCS, pp. 487-504, Springer, 2011.
[10] Z. Brakerski, V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE", Electronic Colloquium on Computational Complexity (ECCC) 18: 109, 2011.
[11] Z. Brakerski, V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages. CRYPTO 2011, pp.505-524.
[12] Z. Brakerski, C Gentry, V. Vaikuntanathan, "Fully homomorphic encryption without bootstrapping", Electronic Colloquium on Computational Complexity (ECCC) 18: 111, 2011
[13] M. MikuŠ, "On implementation of the Gentry-Halevi somewhat homomorphic scheme", Proceedings of the International conference on Computers and Computing, ICCC'11, pp 131-134, 2011.
[14] H. Perl, M. Brenner, M. Smith, "An Implementation of the fully homomorphic Smart-Vercauteren cryptosystem", POSTER,18th ACM Conference on Computer and Communications Security CCS-2011.
[15] P. Scholl, N.P. Smart, "Improved key generation for Gentry's fully homomorphic encryption Scheme", Cryptology ePrint Archive: Report 2011/471, http://eprint.iacr.org/2011/471
[16] C. Gentry, S. Halevi, "Fully homomorphic encryption without Squashing using depth-3 arithmetic circuits", Cryptology ePrint Archive: Report 2011/279. http://eprint.iacr.org/2011/279
[17] K. Lauter, M. Naehrig V. Vaikuntanathan. "Can Homomorphic Encryption be practical?", Cryptology ePrint Archive: Report 2011/405 http://eprint.iacr.org/2011/405
[18] C. Gentry, S. Halevi, "Implementing Gentry's fully homomorphic encryption scheme. EUROCRYPT'11, Vol. 6632 of LNCS, pp. 129–148, Springer, 2011.
[19] V. Vaikuntanathan, "Computing Blindfolded: New developments in Fully homomorphic encryption", Manuscript at http://www.cs.toronto.edu/~vinodv/FHE-focs-survey.pdf
[20] C. Fontaine, F. Galand, "A survey of Homomorphic Encryption for nonspecialists," EURASIP Journal on Information Security, Vol. 2007, Article ID 13801, 2007.
[21] Jibang Liu, Yung-Hsiang Lu, Cheng-Kok Koh, "Performance Analysis of Arithmetic Operations in Homomorphic Encryption" ECE Technical Reports. Paper 404, 2010 http://docs.lib.purdue.edu/ecetr/404
[22] N. Howgrave-Graham. "Approximate Integer Common Divisors", CaLC 2001, Vol. 2146 of LNCS, pp 51–66, Springer, 2001.
[23] Y. Chen, P. Q. Nguyen, "Faster algorithms for approximate common divisors: Breaking fully homomorphic-encryption challenges over the integers", Cryptology ePrint Archive, Report 2011/436, http://eprint.iacr.org/2011/436.
[24] M. Briggs, "An Introduction to the General Number Field Sieve", Master's Thesis, Virginia Tech, April 1998. http://scholar.lib.vt.edu/theses/available/etd-32298-93111/.
[25] H. Lenstra. "Factoring Integers with Elliptic Curves". Annals of Mathematics, 126(1987), pp 649-673.
[26] D.K. Rappe, "Homomorphic cryptosystems and their applications", Doctoral dissertation thesis, University of Dortmund, Germany, 2004. www.rappe.de/doerte/Diss.pdf
[27] D E. Knuth. The art of computer programming, seminumerical algorithms, Vol 2, Addison-Wesley, 3rd edition, 1997.
[28] H. Jing-Li, Y. Ming, W. Zhao-Li, "Fully homomorphic encryption scheme extended to large message space", International Conference on Instrumentation, Measurement, Computer, Communication and Control, pp.533-536, IEEE, 2011.
[29] Y. Govinda Ramaiah, G. Vijaya Kumari, "Efficient public key generation for homomorphic encryption over the integers", 3rd International Conference on Advances in Communication, Network and Computing, CNC-2012, Janahan Lal Stephen (Ed.), LNICST, pp. 262–268, Springer, 2012.
[30] J. S. Coron, D. Naccache, M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers", Cryptology ePrint Archive, Report 2011/440, 2011.
[31] Hao-Miao Yang, Qi XIA, Xiao-fen Wang, Dian-hua Tang, "A New Somewhat Homomorphic Encryption Scheme over Integers", International Conference on Computer Distributed Control and Intelligent Environmental Monitoring CDCIEM-2012, pp.61-64
[32] Y. Govinda Ramaiah, G. Vijaya Kumari, "Efficient Public key Homomorphic Encryption over Integer Plaintexts", Third International Conference on Information Security and Intelligent Control, pp. 126-131, IEEE, 2012
[33] V. Shoup. NTL: A Library for doing Number Theory. http://shoup.net/ntl/ , Version 5.5.2, 2010.

DOI: 01.IJNS.3.4.1071